Dan_Crouthamel_HW3

June 1, 2021

1 Dan Crouthamel – SMU NLP Course — Homework 3

1.1 Assignment Objectives

1. Compare your given name with your nickname (if you don't have a nickname, invent one for this assignment) by answering the following questions:

- What is the edit distance between your nickname and your given name?
- What is the percentage string match between your nickname and your given name?

  Show your work for both calculations

2. Find a friend (or family member or classmate) who you know has read a certain book. Without your friend knowing, copy the first two sentences of that book. Now rewrite the words from those sentences, excluding stop words. Now tell your friend to guess which book the words are from by reading them just that list of words. Did you friend correctly guess the book on the first try? What did he or she guess? Explain why you think you friend either was or was not able to guess the book from hearing the list of words

3. Run one of the stemmers available in Python. Run the same two sentences from question 2 above through the stemmer and show the results. How many of the outputted stems are valid morphological roots of the corresponding words? Express this answer as a percentage.

1.2 Solution

1.2.1 Library Imports

[1]:
```
import nltk
from nltk import edit_distance
from nltk.tokenize import word_tokenize
from difflib import SequenceMatcher
import re
```

1.2.2 Question 1

To compute the edit distance metric, we'll use the edit_distance metric function, which computes the Levenshtein edit distance between two strings. The edit distance is the number of characters that need to be substituted, inserted, or deleted, to transform one string into another.

https://www.nltk.org/api/nltk.metrics.html

My given name is Daniel, but growing up I went by the name of Danny. Today most people call me Dan. Below we can see that the edit distance between my full name and either nickname is 3.

[2]:
```
fullName = 'Daniel'
nick1 = 'Danny'
nick2 = 'Dan'

print('The edit distance between Daniel and Danny:',␣
 ↪edit_distance(fullName,nick1))
print('The edit distance between Daniel and Dan:',␣
 ↪edit_distance(fullName,nick2))
```

```
The edit distance between Daniel and Danny: 3
The edit distance between Daniel and Dan: 3
```

To compute the percentage string match, I used the SequenceMatcher function based on info from the below article.

https://stackoverflow.com/questions/17388213/find-the-similarity-metric-between-two-strings

[3]:
```
print('Percent String Match between Daniel and Danny -> {0:.0%}'.
 ↪format(SequenceMatcher(None, fullName, nick1).ratio()))
print('Precent String Match between Daniel and Dan -> {0:.0%}'.
 ↪format(SequenceMatcher(None, fullName, nick2).ratio()))
```

```
Percent String Match between Daniel and Danny -> 55%
Precent String Match between Daniel and Dan -> 67%
```

1.2.3 Question 2

The book I choose from my fiancé's reading list was "The Kite Runner", by Khaled Hosseini. The first two sentences are below.

"I became what I am today at the age of twelve, on a frigid overcast day in the winter of 1975. I remember the precise moment, crouching behind a crumbling mud wall, peeking into the alley near the frozen creek."

[4]:
```
kite_book = "I became what I am today at the age of twelve, on a frigid␣
 ↪overcast day in the winter of 1975. I remember the precise moment, crouching␣
 ↪behind a crumbling mud wall, peeking into the alley near the frozen creek."

# Remove punctuation
# https://www.geeksforgeeks.org/python-remove-punctuation-from-string/
kite_book = re.sub(r'[^\w\s]+', '', kite_book)

# Define stop words
stopwords = nltk.corpus.stopwords.words('english')

# Tokenize the sentences
kite_tokens = word_tokenize(kite_book)
```

```
# Remove stop words from sentences
kite_tokens_no_stop_words = [w for w in kite_tokens if w.lower() not in␣
 ↪stopwords]

print("")
print("Original Words")
print(kite_tokens)

print("")
print("First two sentences with stop words removed.")
print(kite_tokens_no_stop_words)
```

```
Original Words
['I', 'became', 'what', 'I', 'am', 'today', 'at', 'the', 'age', 'of', 'twelve',
'on', 'a', 'frigid', 'overcast', 'day', 'in', 'the', 'winter', 'of', '1975',
'I', 'remember', 'the', 'precise', 'moment', 'crouching', 'behind', 'a',
'crumbling', 'mud', 'wall', 'peeking', 'into', 'the', 'alley', 'near', 'the',
'frozen', 'creek']

First two sentences with stop words removed.
['became', 'today', 'age', 'twelve', 'frigid', 'overcast', 'day', 'winter',
'1975', 'remember', 'precise', 'moment', 'crouching', 'behind', 'crumbling',
'mud', 'wall', 'peeking', 'alley', 'near', 'frozen', 'creek']
```

My girlfriend, Joy, did guess the book on the first try, and was surprised that she did. She couldn't say why, other than maybe becuase the story is about a little boy. I've never read the book, but did see the movie years ago.

Joy's native language is Korean, and after discussing the Stop Words concept with her she mentioned that they don't really exist in Korean. She feels that English is a very inefficient language. Perhaps she is correct :) I wonder if there is a language that has symbols that can encapsulate an entire paragraph? Something that you can experience in a single thought or moment, rather than sounding out a bunch of syllables and taking too much time.

1.2.4 Question 3

For question 3, I'll use the Porter stemmer. We'll then compare that output with nltk.corpus.words, to see if it's a valid word. I'm not sure those if that really answers the question that is is a valid morpholicial root, but it's the best I can think of for now. If there is anything better, please let me know!

https://www.nltk.org/api/nltk.stem.html

With the approach outlined above, we see that the Percentage of stems that are valid words is 77%.

[5]:
```
porter = nltk.PorterStemmer()

new_text = [porter.stem(w) for w in kite_tokens_no_stop_words]
```

```
print(new_text)
```

```
['becam', 'today', 'age', 'twelv', 'frigid', 'overcast', 'day', 'winter',
'1975', 'rememb', 'precis', 'moment', 'crouch', 'behind', 'crumbl', 'mud',
'wall', 'peek', 'alley', 'near', 'frozen', 'creek']
```

[6]:
```
# Import valid words here to compare our stems against
from nltk.corpus import words

valid_words = [word for word in new_text if word in words.words()]

print("Percentage of stems that are valid words -> {0:.0%}".
 ↪format(len(valid_words)/len(new_text)))
```

```
Percentage of stems that are valid words -> 77%
```